This document contains the essential properties of room and player objects that can be used to customize/develop the client side application.

**ROOM MODEL**

The server maintains all the active rooms as an array data structure.

Each room is an object with these useful properties:

{

     isFull: true/false,

     roomType: 'TWO\_PLAYER'/'THREE\_PLAYER'/'FOUR\_PLAYER',

     gameType: 3/4/5,

     maxPlayers: 2/3/4,

     roomId: TWO\_PLAYER-1',

     players: [ array of player objects ],

     private: true/false,

     board: ['', '', '', '', '', '', '', '', ''],

     timeout: 10 seconds,

     gameStarted: false,

}

Description of room properties:

**1. isFull**

This property is true if the all the players have joined the room and there is no more available space. Useful for private room joining where you can check if the room to join is full or empty.

**2. roomType**

This property determines the type of the room. It is used internally by the server. Here are its possible values:

For 3x3 value is: “TWO\_PLAYER”

For 4x4 value is: “THREE\_PLAYER”

For 5x5 value is: “FOUR\_PLAYER”

**3. gameType**

This property deteremines the type of the game.

You can use game type to directly know whether the room is a 3x3 or 4x4 or 5x5 game.

For 3x3 value is: 3

For 4x4 value is: 4

For 5x5 value is: 5

**4. maxPlayers**

This property stores the maximum number of players that a room can accommodate. Possible values: 2, 3, 4 depending on the type of the game.

**5. roomId**

roomId is a unique identifier for the room and used across different places in the code.

**6. players**

Stores the array of player objects in the room.

**7. private**

This property is true if the room is a private one and not a public one. Using the join room feature, only private rooms can be joined using the roomCode/Id.

**8. board**

Stores the board as an array of strings. It is used by the server to maintain the game status like losing, winning, etc.

Eg: board: [‘X’, ‘O’, ‘’, ‘’, ‘X’, ‘’, ‘’, ‘’, ‘O’]

**9. timeout**

Stores the timer in seconds. Used to switch turns to other players within 10 seconds. If no move is made in 9 seconds, then after the 9th second, the room marks/moves on behalf of that player.

**10. gameStarted**

Initially false, but once the game has started, it is set to true.

**PLAYER MODEL**

Every room is an object and has a players property which is an array of player objects.

Each player object has these useful properties:

{

**name**: ‘player-name’,

**socketId**: ‘socket.id’,

**roomId**: ‘id-of-the-room’,

**turn**: true/false,

**symbol**: ‘x’/’y’/’o’/’t’,

**robot**: true/false

}

**1. name**

Name is just one of the property that the client can share with the server. We can extend to other properties like name, age, etc. player details.

**2. socketId**

socketId is a unique id that identifies the player’s socket connection.

**3. roomId**

Player stores the roomId where he is currently joined. It can be either private room or public room.

**4. turn**

This property is set to true by the server when it the player’s turn comes, which the client side app can use to allow a player to make a move/mark on the board.

**5. symbol**

When a player makes a move/mark on the board, the symbol should be used to display the player sign (X or O or Y or T) on the board.

**6. robot**

This property of the player becomes true when the player leaves the game. So the robot continues to play when the player leaves the game, so it doesn’t effect the other players.